Hybrid Cuckoo Search Algorithm based on Powell Search for Constrained Engineering Design Optimization

WEN LONG
Guizhou University of Finance and Economics
Guizhou Key Laboratory of Economics System Simulation
Guiyang 550004
CHINA
lw084601012@gmail.com

JIANJUN JIAO
Guizhou University of Finance and Economics
School of Mathematics and Statistics
Guiyang 550004
CHINA
dragon638@126.com

Abstract: Cuckoo search (CS) has been recently proposed as a population-based optimization algorithm and it is has so far been successfully applied in a variety of fields. An efficient hybrid cuckoo search algorithm (HCSA) based on Powell direct search method is proposed for solving constrained engineering design optimization problems. The inertia weight of Levy flights is introduced to balance the ability of global and local search. The Powell local search technique is used to improve the best approximation found by the cuckoo search algorithm. The hybridization of cuckoo search algorithm and Powell search method may provide a more effective trade-off between exploitation and exploration of the search space. The proposed hybrid method is tested based on three widely used benchmark constrained functions and three well-known constrained engineering design optimization problems, which shows that HCSA is of better or competitive performances when compared with several existing algorithms.

Key–Words: cuckoo search algorithm, Powell search, constrained optimization, engineering design optimization.

1 Introduction

Many real-world design optimization problems in engineering are constrained optimization problems, such as welded beam design [1], pressure vessel design [2], and tension/compression spring design [3] and so on. The general constrained optimization problem with equality, inequality is defined as

$$\begin{array}{ll} \min & f(\vec{x}),\ \vec{x}=(x_1,x_2,...,x_n) \\ s.t. & g_j(\vec{x})=0, j=1,2,...,p, \\ & g_j(\vec{x})\le 0, j=p+1,...,m, \\ & l_i\le x_i\le u_i, i=1,2,...,n. \end{array} \tag{1}$$

where $f(\vec{x})$is an objective function, $g_j(\vec{x})=0$ and $g_j(\vec{x})\le 0$ are known as equality and inequality constraints, respectively. p is the number of equality constraints and $m-p$ is the number of inequality constraints, l_i and u_i are the lower bound and the upper bound of x_i, respectively.

The constrained optimization problem (1) is often highly nonlinear, involving many different design variables under complex constraints. Therefore, classical gradient-based optimization approaches have difficulties to handle problem (1). Compared with gradient-based optimization approaches, modern meta-heuristic algorithms are population-based global search techniques, not sensitive to the characteristics of the problems and easy to implement. Due to these advantages, meta-heuristics algorithms have been widely applied to solve constrained optimization problems during the two last decades [4-6].

Cuckoo search (CS) algorithm has been recently developed by Yang and Deb [7] as a meta-heuristics population based optimization method. This algorithm is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Levy flight behavior of some birds and fruit flies. The preliminary studies show that the CS algorithm is very promising and could outperform existing algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO) [7]. Moreover, CS algorithm has shown good performance both on benchmark unconstrained functions and real-world problems [8-9].

Although the CS algorithm is good at exploring the search space and locating the region of global minimum, it is slow at exploiting the solutions [10]. Recently, hybridization of meta-heuristic search algorithms with either deterministic or random local search has appeared in the literature [11]. The hybridization way provides a more effective trade-off between exploitation and exploration of the search space. To the best of our knowledge, the hybridization of CS algorithm and Powell direct search method has not been attempted yet. In this paper, an effective hybrid CS algorithm based Powell search approach is proposed to solve constrained numerical and engi-

Algorithm 1. Cuckoo search algorithm

```
Objective function $f(\vec{x})$, $x = (x_1, x_2, \cdots, x_n)$
Generation $t = 1$;
Initial a population of $n$ host nests $x_i$,($i = 1, 2, \cdots, n$);
**While** ($t <$ Maximum Generation) or (stop criterion)
    Get a cuckoo (say $i$) randomly by Lévy flights;
    Evaluate fitness for cuckoo $F$;
    Choose a nest among $n$ (say $j$) randomly;
    **If** ( $F_i > F_j$) **then**
    Replace $j$ by the new solution;
    **End if**
    Abandon a faction ($P_\alpha$ ) of worse nests and build new ones;
    Keep the best solutions (or nests with quality solutions)
    Rank the solutions and find the current best;
    Update the generation number $t = T + 1$ ;
**End while**
```

Figure 1: Pseudo code of the cuckoo search algorithm.

neering design optimization problems. The proposed algorithm is applied to three classical benchmark constrained functions and three benchmark problems in engineering, reported in the specialized literature. The performance of the proposed algorithm is further compared with various approaches, state of the art representatives in this area.

2 Cuckoo Search Algorithm

In CS algorithm, a potential solution corresponds to a cuckoo egg. CS algorithm is introduced in three idealized rules [7]: 1) Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest. 2) The best nest with high quality of eggs (solutions) will carry over to the next generation. 3) The number of available host nests is fixed, and a host can discover an alien egg with a probability . If cuckoo egg is disclosed by the host, it may be thrown away, or the host may abandon its own nest and commit it to the cuckoo intruder.

Based on these three rules, the basic steps of CS can be summarized as the pseudo code shown in Fig. 1.

When generating new solution for $\vec{x}^{t+1}$, say cuckoo i , a Levy flight is performed

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \alpha \oplus \text{Levy}(\lambda) \tag{2}$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interest. The production $\oplus$ means entry-wise multiplications. Levy flights essentially provide a random walk while their random steps are drawn from a Levy distribution for large steps

$$\text{Levy}u = t^{-\lambda} \tag{3}$$

where $1 \leq \lambda \leq 3$. Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail.

3 Hybrid CS Algorithm

3.1 Inertia Weight

In general, in nature, the foraging path of an animal is effectively a random walk because the next move is based on the current location/state and the transition probability to the next location. Thus, in CS algorithm, Levy flights are a random walk whose step length is drawn from the Levy distribution. To balance the global and local ability of CS algorithm, in this work, when generating new solution $\vec{x}^{t+1}$ for, say cuckoo i , a Levy flight integrating with the inertia weight, w_{iter} , which controls the search ability is performed

$$\vec{x}_i^{t+1} = w_{iter} \cdot \vec{x}_i^t + \alpha \oplus \text{Levy}(\lambda) \tag{4}$$

While introducing the concept of inertia weight w_{iter}, Abdul-Rani et al. [12] observed that a reasonable choice for w_{iter} should linearly decreased from a relatively large value to a small value through the course so that the CS algorithm had a better performance compared with fixed w_{iter} settings. Technically, the larger w_{iter} has greater global search ability whereas the small w_{iter} has greater local search ability. In this study, based on (4), the inertia weight w_{iter} as a nonlinear function of the present iteration number (iter) at each time step. The proposed adaptation of w_{iter} is given as [13]:

$$w_{iter} = w_{initial} \times u^{-iter} \tag{5}$$

where $w_{initial}$ is the initial inertia weight value selected in the range [0,1] and u is a constant value in the range [1.0001,1.005]. In the experiments conducted in [13], u is set to 1.0002 and the performance of the algorithm is evaluated for different values of $w_{initial}$. A larger inertia weight facilitates global exploration and a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area.

3.2 Powell search method

Powell's search is an extension of basic pattern search method. It is based on conjugate direction method and is expected to speed up the convergence of nonlinear objective functions. A conjugate direction method minimizes a quadratic function in a finite number function; it can be minimized in few steps using Powell's method [14].

In Powell's method [15], the initial step is to set search coordinate directions.

$$S_g^h = \begin{cases} 1, & g = h \\ 0, & g \neq h \end{cases} \quad (g, h = 1, 2, ..., n) \tag{6}$$

A step length is randomly generated by implying Eq. (7) to minimize function.

$$\lambda_g^* = \lambda_g^{\min} + (\lambda_g^{\max} - \lambda_g^{\min}) rand \; (g = 1, 2, ..., n) \tag{7}$$

The decision variable X_g is modified once along the coordinate direction (h) as

$$X_g = X_g + \lambda_g^* S_g^h \; (g = 1,\, 2, ..., n) \tag{8}$$

Decision variable is updated only if the new value of decision variable optimizes the objective function. This process continues for all "n" coordinate directions. For next cycle of optimization, pattern search direction is obtained as

$$S_g^h = X_g - Z_g \; (g, h = 1, 2, ..., n) \tag{9}$$

In addition one of the coordinate directions discarded in favor of the pattern direction as

$$S_g^m = S_g^h \; (g = 1,\, 2, ..., n\, ;\; h = n + 1) \tag{10}$$

This process continues until all the coordinate directions have been discarded and whole procedure will restart again along one of the coordinate direction. Finally, updating process continues until the Powell's method has been reached to maximum set iterations.

The flow chart of the Powell's search method is shown in Fig.2.

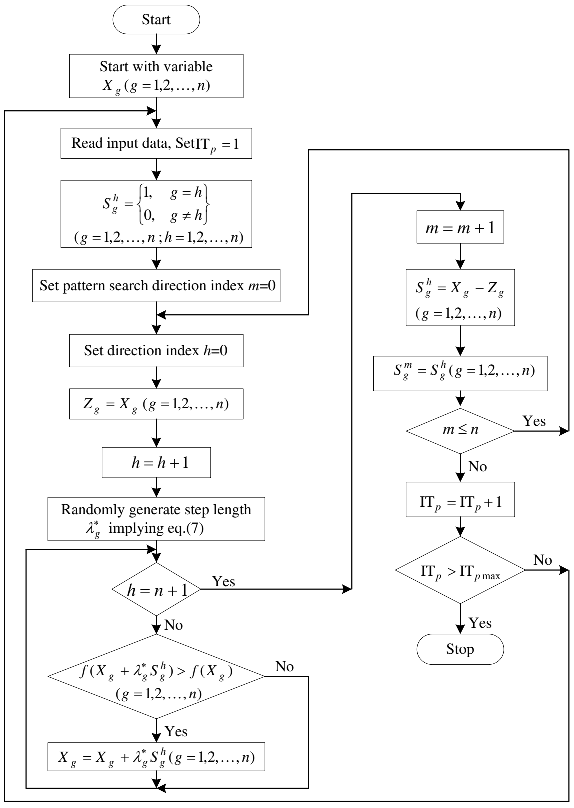
Start
Start with variable $X_g\,(g=1,2,\ldots,n)$
Read input data, Set $IT_p = 1$
$S_g^h = \begin{cases}1, & g=h\\0, & g\neq h\end{cases}$ $(g=1,2,\ldots,n\,;h=1,2,\ldots,n)$
Set pattern search direction index m=0
Set direction index h=0
$Z_g = X_g\,(g=1,2,\ldots,n)$
$h = h+1$
Randomly generate step length λ_g^* implying eq.(7)
$h = n+1$
Yes
No
$f(X_g+\lambda_g^* S_g^h) > f(X_g)$ $(g=1,2,\ldots,n)$
No
Yes
$X_g = X_g + \lambda_g^* S_g^h (g=1,2,\ldots,n)$
$m = m+1$
$S_g^h = X_g - Z_g$ $(g=1,2,\ldots,n)$
$S_g^m = S_g^h (g=1,2,\ldots,n)$
$m \le n$
Yes
No
$IT_p = IT_p + 1$
$IT_p > IT_{p\max}$
No
Yes
Stop

Figure 2: Flow chart of the Powell's search method.

3.3 Constraint Handling Method

It is necessary to note that cuckoo search algorithm is unconstrained optimization methods that need additional mechanisms to deal with constraints when solving constrained optimization problems. Coello [16] provided a comprehensive survey of the most popular constraint-handling techniques currently used with evolutionary algorithms and grouped them into five categories: 1) penalty functions; 2) special representation and operators; 3) repair algorithms; 4) separate objective and constraints; and 5) hybrid methods.

One of the most popular constraint handling techniques used with meta-heuristics algorithms has been Stochastic Ranking (SR) [4]. SR adopts a rank selection mechanism that tries to balance the influence of considering either the objective function value or the degree of violation of the constraints (a parameter called p_f is adopted for this sake). Thus, the constraint handling technique adopted in this work is the set of SR proposed by Runarsson and Yao [4].

3.4 The Framework of Proposed algorithm

After explaining the main elements of hybrid CS algorithm, the framework of HCSA is clearly illustrated as follows. Firstly, randomly generation an initial population of size from the decision space. In each generation, it exploits the property of CS algorithm, and then Powell's pattern search method has been used further enhance the performance, by examining neighboring solutions. The procedure is repeated to update the population in every iteration. Detailed steps of the hybrid CS algorithm are given in the following:

Step 1: Randomly generate an initial population from the decision space.

Step 2: Run modified CS algorithm to generate new solutions.

Step 3: Evaluate the population based on SR technique, rank the solutions and find the current best of the CS population (best solution so far).

Step 4: Apply the Powell's search method on the current best to improve and update this solution.

Step 5: Check convergence condition. If not satisfied, go to Step 2.

4 Experiments and Comparisons

In this section, the proposed method, HCSA is tested on three classical constrained functions and its performance is compared with the representative metaheuristics algorithms. Subsequently, it is applied to three benchmark constrained engineering design optimization problems.

For each testing problem, the parameters of the HCSA are set as follows: population size $N = 50$, $\alpha = 1$, $P_a = 0.25$, $w_{initial} = 0.3$, $P_f = 0.45$. For test functions and engineering design problems, the maximum number of iterations is set to 3000 and 1000, respectively. For each problem, 30 independent runs are carried out.

4.1 Constrained Test Functions

In order to validate the performance of our proposed approach, we adopted three standard test functions form the specialized literature on evolutionary constrained optimization [4].

4.1.1 The First Constrained Function

The first constrained function is a minimization problem with five design variables and six inequality constraints, and this function can be stated as

$$\min \quad f_1(\vec{x}) = 5.357854x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$$s.t. \quad g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$

$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

where $78 \leq x_i \leq 102$, $33 \leq x_2 \leq 45$, and $27 \leq x_i \leq 45 (i = 3, 4, 5)$. The optimum solution is

$$x^* = (78, 33, 29.9953, 45, 36.7758)$$

where the optimal objective function value $f(x^*) = -30665.539$.

This function was previously solved using mine blast algorithm (MBA) [17], particle swarm optimization (PSO) [17], differential evolution (DE) [18], changing range genetic algorithm (CRGA) [19], cultural algorithms with evolutionary programming (CAEP) [20], harmony search (HS) [21], and genetic algorithm (GA) [22]. The statistical results of optimization for seven algorithms including HCSA are shown in Table 1. Table 2 compares the results for HS, GA and HCSA with optimal solution.

Table 1: Comparison of statistical results for various optimizers for first constrained function.

Method	Best	Mean	Worst	St.dev
SR	-30665.539	-30665.539	-30665.539	2.00E-05
PSO	-30663.856	-30570.929	-30252.326	8.10E+01
MBA	-30665.539	-30665.518	-30665.330	5.08E-02
DE	-30665.539	-30665.536	-30665.509	5.07E-03
CRGA	-30665.520	-30664.398	-30660.313	1.60E+00
CAEP	-30665.500	-30665.200	-30662.500	9.30E+00
HCSA	-30665.539	-30665.538	-30665.537	1.65E-07

As shown in Table 1, with respect to PSO, CRGA and CAEP, HCSA finds better "best", "mean", "worst", and "standard deviation" values for first constrained function. Compare with MBA and DE, HCSA provides similar "best" results and better "mean", "worst", and "standard deviation" values for first constrained function. Figure 3 shows the function values versus the number of iterations for first constrained function. By observing Fig. 3, HCSA converged to near optimum point in the early iterations.

4.1.2 The Second Constrained Function

The second constrained function is a minimization problem with seven design variables and four inequal-

Table 2: Comparison of the best solution given by different algorithms for first constrained function.

	HS	GA	HCSA
x_1	78.0	80.39	78.000000
x_2	33.0	35.07	33.000000
x_3	29.995	32.05	29.995256
x_4	45.0	40.33	45.000000
x_5	36.776	33.34	36.775811
$g_1(\vec{x})$	4.34E-05	-0.343809	-2.29E-07
$g_2(\vec{x})$	-92.000043	-91.656190	-92.000000
$g_3(\vec{x})$	-11.15949	-10.463103	-11.159500
$g_4(\vec{x})$	-8.840510	-9.536896	-8.840500
$g_5(\vec{x})$	-5.000064	-4.974473	-5.000000
$g_6(\vec{x})$	6.49E-05	-0.025526	2.7798E-07
$f(\vec{x})$	-30665.500	-30005.700	-30665.539

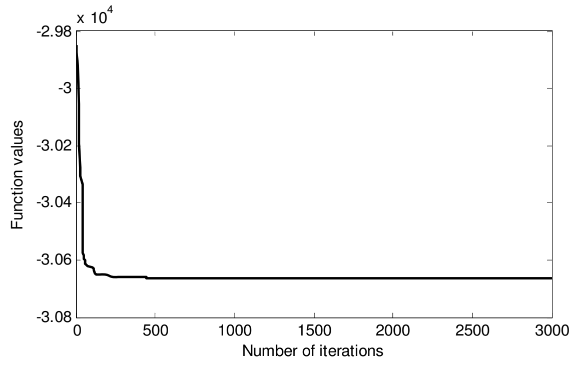

Figure 3: Function values versus number of iterations for first constrained function.

ity constraints, and this function can be stated as small

$$\begin{aligned} \min \quad f_2(\vec{x}) &= (x_1-10)^2+5(x_2-12)^2+x_3^4 \\ &\quad +3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4 \\ &\quad -4x_6x_7-10x_6-8x_7 \\ s.t. \quad & \\ g_1(\vec{x}) &= -127+2x_1^2+3x_2^4 \\ &\quad +x_3+4x_4^2+5x_5 \le 0 \\ g_2(x) &= -282+7x_1+3x_2+10x_3^2 \\ &\quad +x_4-x_5 \le 0 \\ g_3(\vec{x}) &= -196+23x_1+x_2^2+6x_6^2 \\ &\quad -8x_7 \le 0 \\ g_4(\vec{x}) &= 4x_1^2+x_2^2-3x_1x_2 \\ &\quad +2x_3^2+5x_6-11x_7 \le 0 \end{aligned}$$

where $-10 \le x_i \le 10(i=1,2,...,7)$. The optimum solution is

$$x^* = \begin{array}{l}(2.330499, 1.951372, -0.4775414, 4.365726, \\ -0.6244870, 1.038131, 1.594227)\end{array}$$

where the optimal objective function value $f(x^*) = 680.6300573$.

This function was previously solved using SR, PSO, MBA, DE, CRGA, and GA. The statistical optimization results for different algorithms are given in Table 3. Table 4 compares the results for HS, MBA and HCSA with optimal solution.

Table 3: Comparison of statistical results for various optimizers for second constrained function.

Method	Best	Mean	Worst	St.dev
SR	680.630	680.656	680.763	3.40E-02
PSO	680.6346	680.9711	684.5290	8.10E-01
MBA	680.6322	680.6620	680.7882	3.30E-02
DE	680.144	680.503	680.771	6.71E-01
CRGA	680.726	681.347	682.965	5.70E-01
GA	680.6303	680.6381	680.6538	6.61E-03
HCSA	680.6300	680.6300	680.6300	4.59E-05

Table 4: Comparison of the best solution given by different algorithms for second constrained function.

	MBA	HS	HCSA
x_1	2.326585	2.323456	2.330493
x_2	1.950973	1.951242	1.951380
x_3	-0.497446	-0.448467	-0.477547
x_4	4.367508	4.361919	4.365712
x_5	-0.618578	-0.630075	-0.624490
x_6	1.043839	1.038660	1.038155
x_7	1.595928	1.605348	1.594233
$g_1(\vec{x})$	1.17E-04	0.208928	9.68E-05
$g_2(\vec{x})$	-252.400363	-252.878859	-252.561695
$g_3(\vec{x})$	-144.912069	-145.123347	-144.878051
$g_4(\vec{x})$	1.39E-04	-0.263414	-4.49E-05
$f(\vec{x})$	680.6322202	680.6413574	680.6300

From Tables 3 and 4, compare with PSO, MBA, DE, CRGA, and GA, HCSA reached better ”best”, ”mean”, ”worst”, and ”standard deviation” results for second constrained function. With respect to SR, HCSA provided similar ”best” values and better ”mean”, ”worst”, and ”standard deviation” results for second constrained function. Figure 4 illustrates the function values with respect to the number of iterations for second constrained function. As shown in Fig. 4, the HCSA algorithm reached close to the optimum value in the early iterations of the algorithm.

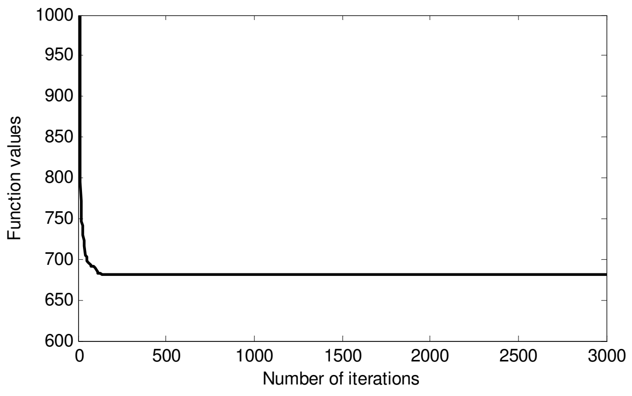

Figure 4: Function values versus number of iterations for second constrained function.

4.1.3 The Third Constrained Function

The third constrained function is minimization problem with two design variables and one equality constraints, and this function can be stated as

$$\begin{array}{ll} \min & f_3(\vec{x}) = x_1^2 + (x_2 - 1)^2 \\ s.t. & h(\vec{x}) = x_2 - x_1^2 = 0 \end{array} \tag{11}$$

where $-1 \leq x_i \leq 1\,(i = 1, 2)$. The optimum solution is $x^* = (-0.70721,\ 0.50000)$ where the optimal objective function value $f(x^*) = 0.749900$.

For this minimization problem, HCSA is compared with six optimizers: SR, PSO, MBA, DE, CRGA, and cultured differential evolution (CDE) [23]. The comparison of statistical results for third constrained function is given in Table 5. Table 6 represents the comparisons between optimal solutions and related design variables for three methods.

Table 5: Comparison of statistical results for various optimizers for third constrained function.

Method	Best	Mean	Worst	St.dev
SR	0.750	0.750	0.750	8.00E-05
PSO	0.750000	0.860530	0.998823	8.40E-02
MBA	0.750000	0.750003	0.750011	3.29E-06
DE	0.74900	0.74900	0.74900	NA
CRGA	0.750	0.752	0.757	2.50E-03
CDE	0.749900	0.757995	0.796455	1.71E-02
HCSA	0.749900	0.750000	0.750000	9.98E-07

As can be seen from Tables 5 and 6, compare with SR, HCSA found better "best" and "standard deviation" results and similar "mean" and "worst" results for third constrained function. With respect to PSO, MBA, and CRGA, HCSA provided similar "best" results and better "mean", "worst", and "standard deviation" results for third constrained function. Compare with CDE, HCSA reached similar "best" results and better "mean", "worst" and "standard deviation" values. Figure 5 depicts the function values versus the number of iterations for the third constrained function. By observing Fig. 5, HCSA reached the near optimal solution in the early iterations of the algorithm.

Table 6: Comparison of the best solution given by different algorithms for third constrained function.

	MBA	CDE	HCSA
x_1	-0.706958	-0.707036	-0.707210
x_2	0.499790	0.500000	0.500240
$h(\vec{x})$	8.82E-15	1.94E-04	9.40E-05
$f(\vec{x})$	0.750000	0.749900	0.749900

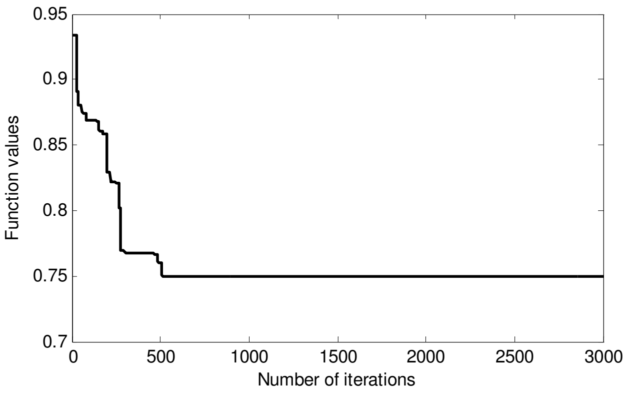

Figure 5: Function values versus number of iterations for third constrained function.

4.2 Engineering Design Problems

4.2.1 Welded Beam Design

This design problem, which has been often used as a benchmark problem, was firstly proposed by Coello [1]. In this problem, the objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress (τ), bending stress (σ) in the beam, buckling load on the bar (P_b), end deflection of the beam (δ), and side constraints. There are four design variables including $x_1(h)$, x_2(l), $x_3(t)$ and $x_4(b)$ as shown in Fig. 6.

The optimization approaches previously applied to welded beam design problem include self adaptive penalty approach (SAPA) [1], MBA, DE, CAEP, genetic algorithms 2 (GA2) [24], and co-evolutionary particle swarm optimization (CPSO) [25]. Table 7

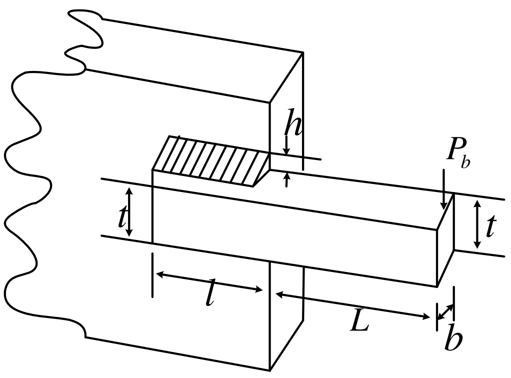

Figure 6: Welded beam design.

presents the results obtained by HCSA and other algorithms. The comparison for the best solution given by SAPA, CPSO and HCSA is shown in Table 8.

Table 7: Comparison of statistical results for various optimizers for welded beam design problem.

Method	Best	Mean	Worst	St.dev
SAPA	1.748309	1.771973	1.785835	1.12E-02
MBA	1.724853	1.724853	1.724853	6.94E-19
DE	1.733461	1.768158	1.824105	2.21E-02
CAEP	1.724852	1.971809	3.179709	4.43E-01
GA2	1.728226	1.792654	1.993408	7.47E-02
CPSO	1.728024	1.748831	1.782143	1.29E-02
HCSA	1.728452	1.724852	1.724852	2.48E-16

As it can be seen in Tables 7 and 8, with respect to MBA, HCSA found a little better result for welded beam design problem. Compare with SAPA, DE, GA2, and CPSO, HCSA provided better ”best”, ”mean”, ”worst”, and ”standard deviation” values. Figure 7 shows the function values versus the number of iterations for welded beam design problem.

4.2.2 Pressure Vessel Design

In the pressure vessel design problem, firstly proposed by Sandgren [1], the objective is to minimize the total cost, including the cost of the material, forming and welding. There are four design variables (see Fig. 8): $x_1(T_s)$, $x_2(T_h)$, $x_3(R)$ and $x_4(L)$.

This problem has been used as a benchmark problem for testing differential optimization methods, such as SAPA, cuckoo search (CS) [8], PSO, MBA, GA2, and CPSO. The obtained statistical results by the considered approaches and HCSA are given in Table 9. The comparison of best solution among several algorithms is given in Table 10. From Tables 9 and 10,

Table 8: Comparison of the best solution given by GA2, CPSO and HCSA for welded beam design problem.

	GA2	CPSO	HCSA
x_1	2.326585	2.323456	2.330493
x_2	1.950973	1.951242	1.951380
x_3	-0.497446	-0.448467	-0.477547
x_4	4.367508	4.361919	4.365712
$g_1(\vec{x})$	1.17E-04	0.208928	9.68E-05
$g_2(\vec{x})$	-252.400363	-252.878859	-252.561695
$g_3(\vec{x})$	-144.912069	-145.123347	-144.878051
$g_4(\vec{x})$	1.39E-04	-0.263414	-4.49E-05
$g_5(\vec{x})$	-252.400363	-252.878859	-252.561695
$g_6(\vec{x})$	-144.912069	-145.123347	-144.878051
$g_7(\vec{x})$	1.39E-04	-0.263414	-4.49E-05
$f(\vec{x})$	1.728226	1.728024	1.724852

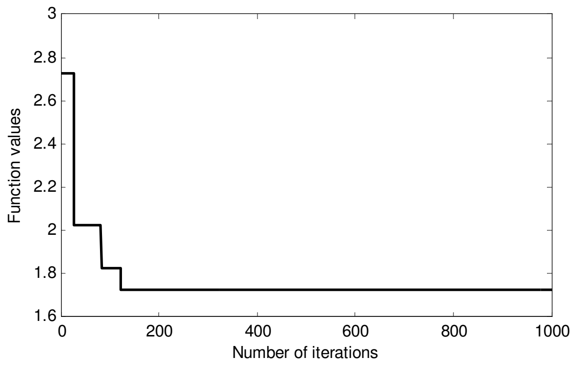

Figure 7: Function values versus number of iterations for welded beam design problem.

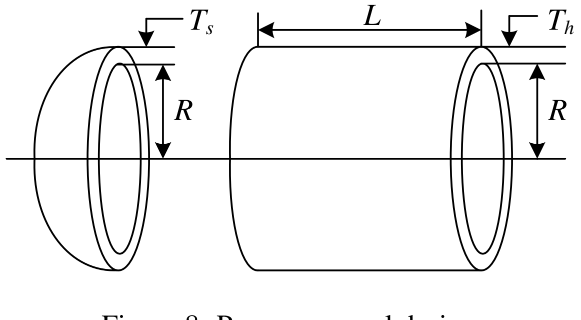

Figure 8: Pressure vessel design.

HCSA outperforms all other algorithms in terms of best solution.

Table 9: Comparison of statistical results for various optimizers for pressure vessel design problem.

Method	Best	Mean	Worst	St.dev
SAPA	6288.7445	6293.8432	6308.4970	7.4133
CS	6059.7143	6447.7360	6495.3470	502.69
MBA	5889.3216	6200.6476	6392.5062	160.34
PSO	6693.7212	8756.6803	14076.324	1492.57
GA2	6059.9463	6177.2533	6469.3220	130.93
CPSO	6061.0777	6147.1332	6363.8041	86.45
HCSA	5885.3328	5895.6407	5912.5280	13.20

Table 10: Comparison of the best solution given by SAPA, CPSO and HCSA for welded beam design problem.

	SAPA	CPSO	HCSA
x_1	0.8125	0.8125	0.7781686414
x_2	0.4375	0.4375	0.3846491626
x_3	40.3239	42.0913	40.319618724
x_4	200.000	176.7465	200.00000000
$g_1(\vec{x})$	-3.42E-02	-1.37E-06	1.1080E-13
$g_2(\vec{x})$	-5.28E-02	-3.59E-04	-9.5479E-14
$g_3(\vec{x})$	-304.4020	-118.7687	-1.9907E-08
$g_4(\vec{x})$	-40.0000	-63.2535	-400.000000
$f(\vec{x})$	6288.7455	6061.0777	5885.3328

Figure 9 shows the function values versus the number of iterations for pressure vessel design problem. As shown in Fig. 9, HCSA converged to near optimum solution at early iterations.

4.2.3 Tension/compression spring design

The tension/compression spring design problem is described in Belegundu [3] for which the objective is to minimize the weight of a tension/compression spring design (see Fig. 10) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the wire diameter $d(x_1)$, the mean coil diameter $D(x_2)$ and the number of active coils $P(x_3)$.

This problem has been solved using SAPA, MBA, DE, CAEP, GA2, CPSO, and improved harmony search (IHS) [26]. The comparison of the statistical

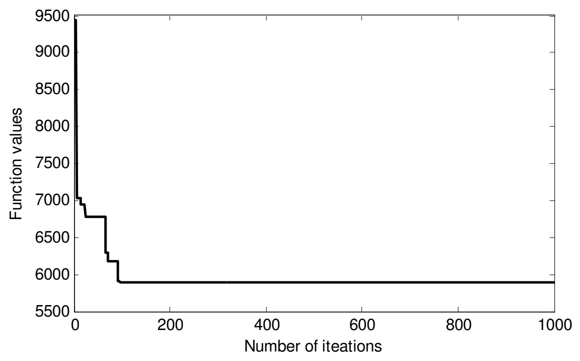

Figure 9: Function values versus number of iterations for pressure vessel design problem.

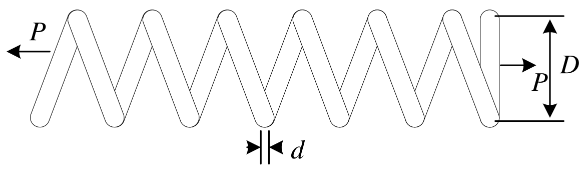

Figure 10: Tension/compression spring design.

results by such algorithms is presented in Table 11. The comparison of best solution obtained by HCSA with previous methods is given in Table 12.

Table 11: Comparison of statistical results for various optimizers for spring design problem.

Method	Best	Mean	Worst	St.dev
SAPA	0.012705	0.012769	0.012822	3.94E-05
MBA	0.012665	0.012713	0.012900	6.30E-05
DE	0.012670	0.012703	0.012790	2.70E-05
CAEP	0.012721	0.013568	0.015116	8.42E-04
GA2	0.012681	0.012742	0.012973	5.90E-05
CPSO	0.012675	0.012730	0.012924	5.20E-04
HCSA	0.012665	0.012677	0.012714	1.77E-05

As can be seen from Tables 11 and 12, compare with SAPA, DE, CAEP, GA2, and CPSO, HCSA finds better "best", "mean", "worst", and "standard deviation" results for tension/compression spring design problem. With respect to MBA, HCSA provides similar "best" results and better "mean", "worst", and "standard deviation" results. Figure 11 depicts the function values with respect to the number of iterations for tension/compression spring design problem. By observing Fig. 11, the function values are reduced to near optimum point at the early iterations.

Based on the aforementioned experimental and comparison results validate that HCSA has the sub-

Table 12: Comparison of the best solution given by three algorithms for spring design problem.

	IHS	CPSO	HCSA
x_1	0.051728	0.05115438	0.051599189
x_2	0.357644	0.34987116	0.354559500
x_3	11.244543	12.0764321	11.41663523
$g_1(\vec{x})$	-8.25E-04	-0.0521995	-6.6481E-07
$g_2(\vec{x})$	-5.28E-02	-3.59E-04	-1.5521E-08
$g_3(\vec{x})$	-4.051306	-3.860150	-4.04949845
$g_4(\vec{x})$	-0.727085	-0.7326496	-0.72922754
$f(\vec{x})$	0.012675	0.012887	0.012665

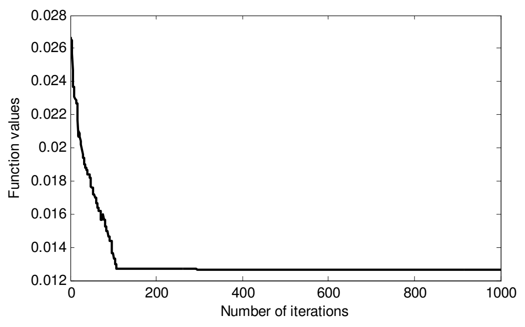

Figure 11: Function values versus number of iterations for spring design problem.

stantial capability in handling various constrained numerical and engineering design problems and its solution quality is quite stable. So, it can be concluded that HCSA is a good alternative for constrained optimization.

5 Conclusion

In this paper, an effective hybrid cuckoo search algorithm based on Powell's search is proposed to solve constrained numerical and engineering design problems. The performance of the proposed algorithm has been investigated by a number of experimental studies. The obtained results show that the proposed algorithm generally offers better solutions than other optimizers considered in this research in terms of objective function values for some problem and the number of iterations for almost every problem.

Acknowledgements: The research was supported by the Science and Technology Foundation of Guizhou Province(grant No. [2013]2082) and supported by the Excellent Science Technology Innovation Talents in University of Guizhou Province(grant No. [2013]140).

References:

[1] C. Coello, Use of a self-adaptive penalty approach for engieering optimization problems, *Comput. Indus.* 41, 2000, pp. 113–127.

[2] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, *ASME J. Mech. Des.* 112, 1990, pp. 223–229.

[3] A. Belegundu, *A study of mathematical programming methods for structural optimization,* Ph.D. thesis. Department of Civil and Environmental Engineering, University of Iowa, Iowa, 1982.

[4] T. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization, *IEEE Trans. Evol. Comput.* 4, 2000, pp. 284–294.

[5] W. Long, X. Liang, Y. Huang and Y. Chen, A hybrid differential evolution augmented Lagrangian method for constrained numerical and engineering optimization, *Comput. Aided Des.* 45, 2013, pp. 1562–1574.

[6] I. Mazhoud, K. Hamou and J. Bigeon, Particle swarm optimization for solving engineering problems: A new constraint-handling mechanism, *Eng. Appl. Art. Intel.* 26, 2013, pp. 1263–1273.

[7] X. Yang and S. Deb, *Cuckoo search via Levy flights,* Proc. of World Congress on Nature and Biologically Inspired Computing 2009, pp. 210–214.

[8] A. Gandomi, X. Yang, and A. Alavi, Cuckoo search algorithm: a meta-heuristic approach to structural optimization problem, *Eng. Comput.* 29, 2013, pp. 17–35.

[9] X. Yang, and S. Deb, Multi-objective cuckoo search for design optimization, *Comput. Oper. Res.* 40, 2013, pp. 1616–1624.

[10] W. Long, X. Liang, Y. Huang and Y. Chen, An effective hybrid cuckoo search algorithm for constrained global optimization, *Neural Comput. Appl.* 2014, in Press, Doi: 10.1007/s00521-014-1577-1.

[11] E. Zahara and Y. Kao, Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems, *Expert Syst. Appl.* 36, 2009, pp. 3880–3886.

[12] K. Abdul-Rain, M. Abd-Malek and N. Siew-Chin, Nature-inspired cuckoo search algorithm for side lobe suppression in a symmetric linear antenna array, *Radio-Eng.* 21, 2012, pp. 865–874.

[13] B. Jiao, Z. Lian and X. Gu, A dynamic inertia weight particle swarm optimization algorithm, *Chaos Solitons Fractals* 37, 2008, pp. 698–705.

[14] S. Rao, *Engineering optimization: theory and practice,* John Wiely and Sons, New York, 1996

[15] N. Narang, J. Dhillon and D. Kothari, Multi-objective fixed head hydrothermal scheduling using integrated predator-prey optimization and Powell search method, *Energy* 47, 2012, pp. 237–252.

[16] C. Coello, Theoretical adn numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art, *Comput. Methods Appl. Mech. Eng.* 191, 2002, pp. 1245–1287.

[17] A. Sadollahm, A. Bahreininejad and H. Eskandar, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, *Appl. Soft Comput.* 13, 2013, pp. 2592–2612.

[18] J. Lampinen, *A constraint handling approach for the differential evolution algorithm,* Proc. of IEEE Congress on Evolutionary Computation, 2002, pp. 1468–1473.

[19] A. Amirjanov, The development of a changing range genetic algorithm, *Comput. Methods Appl. Mech. Eng.* 195, 2006, pp. 2495–2508.

[20] C. Coello and R. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, *Eng. Optim.* 36, 2004, pp. 219–236.

[21] K. Lee and Z. Geem, A new meta-heuristic algorithm for continuous engineering optimization, *Comput. Methods Appl. Mech. Eng.* 194, 2005, pp. 3902–3933.

[22] K. Deb, An efficient constraint handling method for genetic algorithm, *Comput. Methods Appl. Mech. Eng.* 186, 2000, pp. 311–338.

[23] R. Becerra and C. Coello, Cultured differential evolution for constrained optimization, *Comput. Methods Appl. Mech. Eng.* 195, 2006, pp. 4303–4322.

[24] C. Coello and E. Mezura-Montes, Constraint-handling in genetic algorithms through the used of dominance-based tournament selection, *Adv. Eng. Inform.* 16, 2002, pp. 193–203.

[25] Q. He and L. Wang, An effective co-evolutionary particle swarm optimization for engineering optimization problems, *Eng. Appl. Art. Intel.* 20, 2007, pp. 89–99.

[26] M. Mahdavi, M. Fesanghary and E. Damangir, An improved harmony search algorithm for solving optimization problems, *Appl. Math. Comput.* 188, 2007, pp. 1567–1579.